Homework #3

Due: Tuesday, October 28, 1996, at 11:59PM
Points: 130

The UNIX Environment

Submitting: Type your answers into a file called *Answers*, and submit it as described in the **All About Homework** handout. Remember, we'll read it on the DECs, so be sure it can be read on them! (**Note**: if you use another file name, we will deduct 5 points.)

The purpose of these questions is to have you use the UNIX system and environment.

1. (*10 points*) To which group does the file /etc/motd belong?
2. (*10 points*) Give a UNIX command to print dictionary words that contain the vowels a, e, I, o, and u in that order.

   Hint: The file /usr/dict/words is a dictionary of English words.
3. (*10 points*) What does the C library function *system* do?

   Hint: It is described in section 3 of the UNIX Programmers' Manual.

The C Programming Language

Submitting: Put your program into a file called *main.c*, and submit it as described in the **All About Homework** handout. Remember, we'llcompile and run it on the DECs, so be sure it works there! (**Note**: if you use another file name, we will deduct 5 points.)

(*100* points) The purpose of this question is for you to build on your (or my) answer to program 2 to gain experience with pointers and command-line arguments.

Input Specifications

Read from the standard input or from a list of files named on the command line. The input is supposed to be a C program. You need not check that it is, in fact, a legal C program; your program should simply assume it is.

Your program must handle the following command-line option:

–Dsym=val

Predefine the macro sym to have a value of val. If =val is omitted, the symbol is defined as the empty string. Thus, typing

```
main –Dunix=1 –Ddec
```

has the same effect of putting the preprocessor commands

```
#define unix 1
#define dec
```

at the head of each file.

Output Specifications

Write to the standard output. In the first 6 columns of each output line, print the line number followed by a period ".". Print the input line beginning in column 9 (one tab from the left margin). If the input is from a file, print the name of the file followed by a ":" on the line before the first line.

If the line is a preprocessor control line, and the preprocessor directive is **define**, add the name, line number on which the **define** occurs, and definition to a table. When the input is finished, print the table, headed by the title "Table of Defines" beginning on a separate page.

If the macro is defined on the command line (using the –D option), it is to be added to the table and printed, but the line number is to be printed as "*predefined*",

If multiple files are named on the command line, the files are to all be processed first, and then the table printed. Note this means one symbol may have multiple definitions; this is not an error, and the definitions should all appear in the table.

Other Constraints

Put the entries into an array, one definition and one name per entry. (So, you will have to use a structure for this.) The form of each entry is

```
struct defent {
       char *name;
       char *defn;
       char *filename;
       int lineno;
};
```

You must allocate space for the `name` and `defn` fields, but you may assume that no more than 1000 symbols will be defined.

Example Input

```
/* file test1.c */
#define PI 3.14159

extern void square(double *);
void main(void)
{
      double f = PI;
      timespie(&f);
      printf("pi = %f, pi^2*e = %f\n", PI, f);
}


/* file test1-a.c */
#define PI (22/7)
#define E  2.81728
void square(double *d)
{
      return(d * PI * E);
}
```

Example Output

```
test1.c:
     1. /* file test1.c */
     2. #define PI 3.14159
     3.
     4. extern void square(double *);
     5. void main(void)
     6. {
     7.     double f = PI;
     8.     timespie(&f);
     9.     printf("pi = %f, pi^2*e = %f\n", PI, f);
    10. }
test1-a.c:
     1. /* file test2.c */
     2. #define PI (22/7)
```

```
3. #define E  2.81728
4. void timespie(double *d)
5. {
6.     return(d * PI * E);
7. }
```

```
Table of Defines:
name            line file              definition
PI                2  test1.c          3.14159
PI                2  test1-a.c          (22/7)
E                 3  test1-a.c          2.81828
```

Extra Credit

Submitting: Create a C program called *extra.c*. You may copy your *main.c* into it and modify that (in fact, we encourage you to!)

The modification for the extra credit is to print the base name of the file before the line number. The output for the example input would look like this:

```
test1.c:        1. /* file test1.c */
test1.c:        2. #define PI 3.14159
test1.c:        3.
test1.c:        4. extern void square(double *);
test1.c:        5. void main(void)
test1.c:        6. {
test1.c:        7.         double f = PI;
test1.c:        8.         timespie(&f);
test1.c:        9.         printf("pi = %f, pi^2*e = %f\n", PI, f);
test1.c:       10. }
test1-a.c:      1. /* file test2.c */
test1-a.c:      2. #define PI (22/7)
test1-a.c:      3. #define E     2.81728
test1-a.c:      4. void timespie(double *d)
test1-a.c:      5. {
test1-a.c:      6.         return(d * PI * E);
test1-a.c:      7. }
```

Note that the line numbers are all aligned. This requires you to compute the length of the longest base name and format accordingly.

The base name of a file is the last component of the full path name. For example, the base name of the file /usr/include/stdio.h is stdio.h and the base name of the directory /home/bishop is bishop.